LA-UR-03-6274

Approved for public release; distribution is unlimited.

Title:

SEU MITIGATION FOR HALF-LATCHES IN XILINX VIRTEX FPGAS

Author(s):

Paul Stanley Graham, 181856, NIS-3
Michael Paul Caffrey, 111180, NIS-3
Darrel Eric Johnson, 187949, NIS-3/Brigham Young University
Nathaniel Hatley Rollins, Brigham Young University
Michael Joseph Wirthlin, Brigham Young University

Submitted to:

IEEE Transactions on Nuclear Science (journal) in relation to IEEE Nuclear and Space Radiation Effects Conference, July 21-25, 2003, Monterey, CA, USA for

Los Alamos
NATIONAL LABORATORY

SEU Mitigation for Half-Latches in Xilinx Virtex FPGAs

Paul Graham, Michael Caffrey, D. Eric Johnson, Nathaniel Rollins, and Michael Wirthlin

***Abstract*— The performance, in-system reprogrammability, flexibility, and reduced costs of SRAM-based field-programmable gate arrays (FPGAs) make them very interesting for high-speed, on-orbit data processing, but the current generation of radiation-tolerant SRAM-based FPGAs are based on commercial-off-the-shelf technologies and, consequently, are susceptible to single event upset effects. In this paper, we discuss in detail the consequences of radiation-induced single-event upsets (SEUs) in the state of half-latch structures found in Xilinx Virtex FPGAs and describe methods for mitigating the effects of half-latch SEUs. One mitigation method's effectiveness is then illustrated through experimental data gathered through proton accelerator testing at Crocker Nuclear Laboratory at the University of California-Davis. For the specific design and mitigation methodology tested, the mitigated design demonstrated more than an order of magnitude improvement in reliability over the unmitigated version of the design in regards to average proton fluence until circuit failure.**

***Index Terms*— radiation effects, SEUs, FPGAs, proton accelerator, half-latches.**

I. INTRODUCTION

THE performance, in-system reprogrammability, flexibility, and reduced costs of SRAM-based field-programmable gate arrays (FPGAs) make them very interesting for high-speed, on-orbit data processing, but, because the current generation of radiation-tolerant, SRAM-based Xilinx XQVR FPGAs are derived directly from commercial-off-the-shelf (COTS) Virtex versions of the chips, their memory structures are still susceptible to single-event upsets (SEUs) when placed in the space radiation environment. With SRAM-based FPGAs, this susceptibility to SEUs affects more than just the data stored in a design's memory elements; it can disrupt the function of an FPGA design by changing values stored in the *configuration memory*, which defines the function of logic resources as well as the interconnection of these resources. Radiation-induced SEUs can also disturb important circuits on the FPGA which are not directly configurable or visible to FPGA users.

Previous papers [1]–[8] have described the SEU characteristics and mitigation techniques for the configuration and user memory structures on the Xilinx Virtex and XQVR families of FPGAs. Upsets in the configuration memory can be detected through reading the contents of the configuration memory and then comparing it with a known, good state. These upsets can be repaired through refreshing the state of the configuration memory by writing good programming data into the memory. To further mitigate the effects of SEUs in configuration memory while protecting the integrity of on-chip user data, these and other papers have applied traditional logic redundancy techniques such as triple-modular redundancy and error-correcting codes to the designs implemented on SRAM FPGAs.

As indicated above, a portion of the sensitive cross-section for SRAM FPGAs is made up of circuits which may not be directly visible to or controllable by the FPGA user. For instance, the state machines that control accesses to the configuration memory and the programming of the device are sensitive to SEUs. Upsets in these circuits may be easily detected (for instance, the device may become "unprogrammed") or they may result in behavior that may be harder to diagnose and from which it may be hard to gracefully recover. Upsets in these less visible forms of FPGA circuitry illustrate one of the challenges of using COTS-based SRAM FPGAs (or other COTS devices) in the space environment—SEUs may occur in the device which require internal device knowledge to understand and mitigate.

In this paper, we will concentrate on the effects and mitigation of SEUs on one "hidden" structure found in Xilinx Virtex and XQVR FPGAs called the "half-latch". We will first describe in detail what half-latches are and the consequences of SEUs in half-latch structures. We will then describe techniques for mitigating these effects. Finally, we will provide new experimental data that illustrate the effectiveness of one of these mitigation techniques when an FPGA is exposed to proton radiation.

II. HALF-LATCHES AND SEUS

Xilinx Virtex FPGAs use a little-known resource, called the *half-latch*, to generate many of the constant "0" and "1" logic values used internally by Virtex FPGA designs. The presence of the resource in these devices was first mentioned in [6]. By using half-latches to produce constant logic values, Xilinx can avoid using more expensive logic resources, such as look-up tables (LUTs), to generate constant logic values. At an architectural level, half-latches drive many of the internal inputs to I/O, logic, RAM, clocking, and other resources when there are no direct sources for the inputs, i.e., when the inputs are left unconnected. Hence, half-latches are very efficient and ubiquitous sources of constant "0" and "1" values that are available throughout these devices and, as a result, they are

Manuscript received July 22, 2003; revised August 29 , 2003. This work was supported by the Department of Energy through the Deployable Adaptive Processing Systems project at Los Alamos National Laboratory.

P. Graham and M. Caffrey are with Los Alamos National Laboratory.

D. E. Johnson is with both Los Alamos National Laboratory and Brigham Young University.

N. Rollins and M. Wirthlin are with Brigham Young University.

used frequently by the Xilinx implementation tools in most designs. Our experience has been that large Virtex XCV1000 designs commonly use hundreds or thousands of half-latches.

Within Xilinx's *FPGA Editor* tool (which can be used to observe how a design is implemented with FPGA resources), the use of a half-latch is expressed as a constant "0" or "1" input into the multiplexers (or *muxes*) that exist at the inputs of I/O blocks (IOBs), slices, and other resources. In reality, these muxes only have the ability to select between their inputs or inverted versions of their inputs and the constants illustrated in FPGA Editor are values produced by half-latches and, possibly, the selectable inversion of these input muxes. The resource inputs shown in Table I can be driven by half-latches in the Virtex architecture.

TABLE I
LIST OF KNOWN HALF-LATCHES FOR THE VIRTEX FAMILY

Resource	Inputs
BLOCKRAM	WEAMUX, ENAMUX, RSTAMUX, WEBMUX, ENBMUX, RSTBMUX
BSCAN	TDO1MUX, TDO2MUX
CAPTURE	CAPMUX
DLL	RSTMUX
GCLK	CEMUX
IOB/PCIIOB	SRMUX, TRIMUX, TCEMUX, OMUX, OCEMUX, ICEMUX
PCILOGIC	I1MUX, I2MUX
SLICE	BYMUX, BXMUX, CEMUX, SRMUX, F1-F4, G1-G4
STARTUP	GWEMUX, GTSMUX, GSRMUX
TBUF	TMUX, IMUX

In general, the half-latches driving the inputs to the above mentioned muxes are the critical half-latches. Modifications to the constant values at the inputs of these muxes can have serious consequences to the operation of circuits. On the other hand, the unused inputs to the LUTs (F1-F4, G1-G4) are not as critical since the logic functions are encoded redundantly within the LUT such that "0" or "1" values on the "don't care" inputs result in the same output value—a feature of the Xilinx technology mapping tools that we have been able to verify for several designs.

Figure 1 is a simplified illustration of what half-latches are at the circuit level. The half-latch structure is the PMOS transistor (*T3*) and inverter combination between the input mux (labeled *imux* in the figure) and the two NMOS transistors (*T1* and *T2*) that connect to the FPGA's routing resources. The half-latch is designed so that it can hold a logic "1" at its input (node *A*) using the weak PMOS pull-up transistor *T3* when the input NMOS transistors are turned off. When one of the NMOS transistors is on, the weak pull-up transistor *T3* will be out driven by the incoming signals, allowing node *A* to be driven to "0" by signals from the routing network. In this circuit, the mux which follows the half-latch can be used to selectively invert the input to the resource which follows. This means that the half-latch circuit can be used to produce a logic "0" or "1", as needed by the FPGA user's design. Lastly, to ensure proper operation of user designs, all of the half-latches in the device are initialized to the proper state (a "1" at node *A*) when a Virtex FPGA undergoes "full configuration", i.e., when all of the programming data is updated and the device's "start-up" sequence is executed.

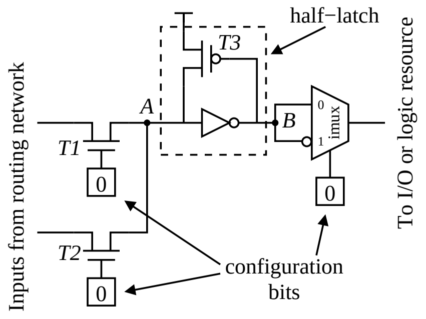


Fig. 1. Half-latch Circuit in the Virtex Architecture

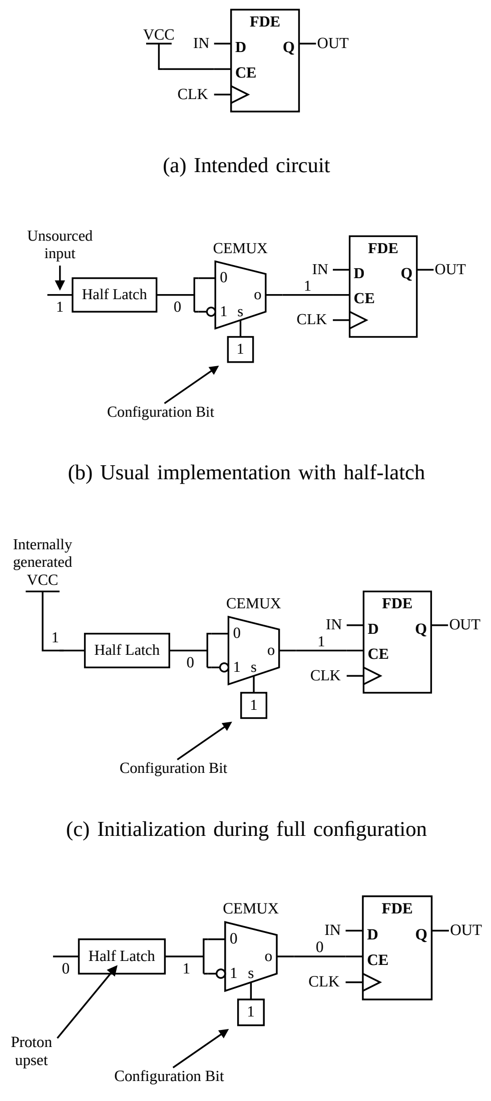
Unfortunately, half-latch resources are susceptible to radiation-induced SEUs when exposed to proton or other forms of radiation—a fact discovered by Xilinx and Los Alamos National Laboratory during their characterization of the Xilinx Virtex and XQVR FPGAs. When upset, the output values of these circuits may remain inverted until the device is fully reprogrammed or another upset occurs. As described in Section IV, we have recently learned that it is also possible for a half-latch to recover to its intended state over time. This is likely due to leakage current through the the weak PMOS pull-up transistor *T3*.

Unlike upsets in the FPGA device's programming data held in the configuration memory, half-latch inversions are not directly observable through reading back the device's programming data since they are not directly initialized or controlled with programming data. Additionally, updating the FPGA's configuration memory with good programming data as is done for SEUs in the configuration memory will not re-initialize any half-latches. The programming mode used to repair a design's configuration memory—a mode called *partial configuration*—does not execute the FPGA's start-up sequence, meaning that the half-latches do not get reinitialized as configuration SEUs are mitigated. Thus, an SEU affecting a half-latch is not as easily detectable or correctable as SEUs in the configuration memory.

A simple example of the half-latch issue is shown in Figure 2. In this case, the designer wanted a flip-flop with the clock enable always asserted, as shown in Figure 2(a). The logic "1" used to drive the clock enable will generally be implemented using a half-latch structure as shown in Figure 2(b). Figure 2(c) illustrates conceptually that the half-latches are initialized during a full configuration of the FPGA. Finally, Figure 2(d) illustrates what happens if the half-latch is upset. In this case, the flip-flop is disabled by the upset and this modification to circuit function cannot be observed through reading back and checking the configuration memory data nor can it be reliably reset except through stopping the design and fully reconfiguring the FPGA.

III. MITIGATION TECHNIQUES

As mentioned in [6], the solution to the hidden half-latch inversion problem is to use explicit constant sources that can



(a) Intended circuit

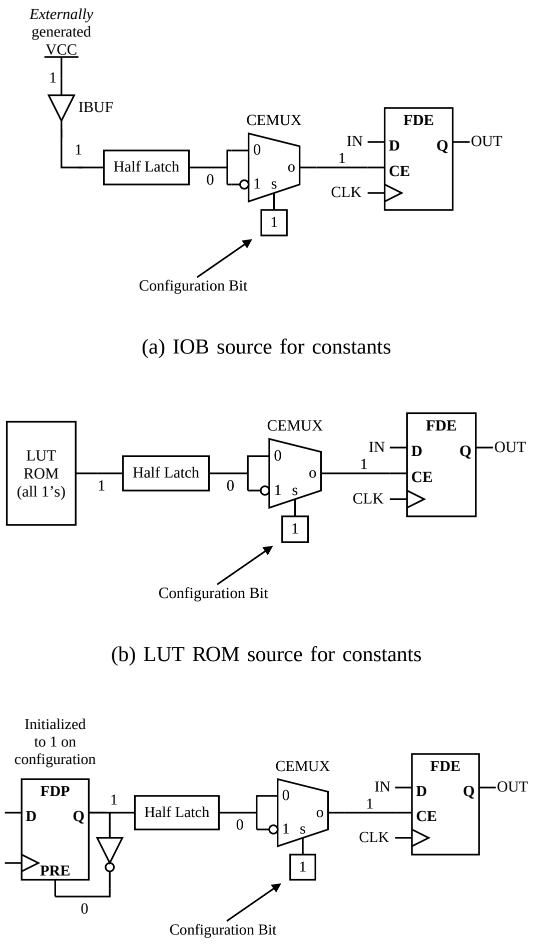
(b) Usual implementation with half-latch

(c) Initialization during full configuration

(d) Circuit with half-latch SEU

Fig. 2. Half-latch illustration showing the intended circuit, the circuit's usual low-level implementation, the initialization of the half-latch at start-up, and the result of a half-latch SEU.

be configured (and fixed) directly through FPGA programming data instead of using the FPGA's implicit constants generated by half-latch resources. As a result, any SEUs that affect constant generation can be both observed and repaired through existing SEU mitigation techniques for configuration bitstreams as described in [2], [3], [5], [7]. Figure 3 illustrates three options for an observable, correctable constant-valued source—an FPGA input pin driven externally by a logic "0" or "1", a LUT ROM filled with "1" values, and a self-correcting flip-flop. While these solutions to the half-latch problem do require additional FPGA resources and use constant sources that are still vulnerable to SEUs, the results are FPGA designs in which logic-constant upsets can be successfully detected and repaired while the designs are still executing as opposed to designs for which half-latch upsets may or may not be detectable and which must be halted by the system to perform a full FPGA reconfiguration to repair any half-latch upsets.



(a) IOB source for constants

(b) LUT ROM source for constants

(c) Flip-flop source for constants

Fig. 3. Observable, correctable constant source alternatives to half-latches

A. Approaches

The half-latch removal process can be performed at several different stages in the design flow, from the HDL or schematic-entry level down to the bitstream level, as Figure 4 shows. As a general comment, the level of design abstraction decreases as a design moves from left to right in the figure.

Approaches that modify designs early in the design flow require careful application to ensure that the design does not use half-latches as constants sources since both synthesis and technology mapping may introduce half-latches into the design implementation due to, among other things, implied constants on FPGA resource inputs or design optimization. At the *Source* stage in the design flow, careful design construction may ensure that half-latches are never used in the design in the first place, but this will require fairly involved and tedious design practices that may be hard to automate. For instance, preventing half-latch usage would require that explicitly generated constant "0" and "1" values be connected throughout the design, as necessary, and that HDL descriptions

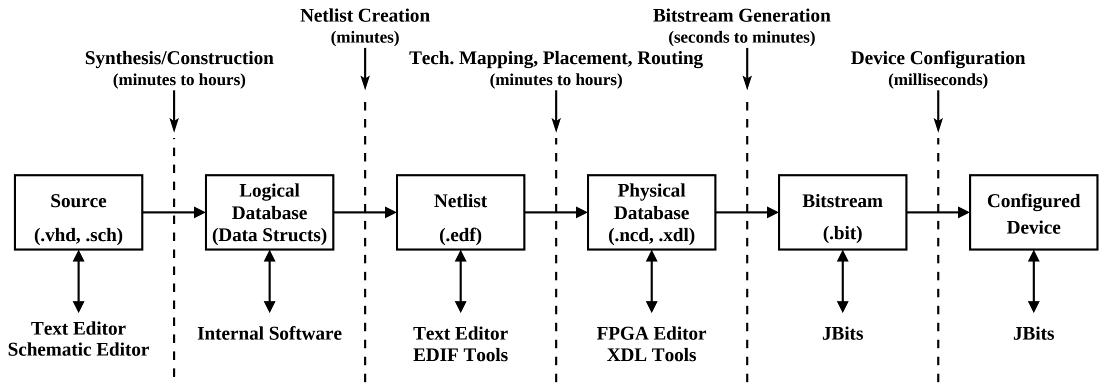


Fig. 4. Xilinx FPGA Design Flow and Design Forms

be carefully structured so that half-latches are not introduced. As an additional complication to these approaches, design practices which worked for older synthesis and technology-mapping tools may not work for new tools as synthesis and technology mapping techniques evolve.

A more interesting alternative to making modifications at the *Source* stage is to perform design primitive replacement at the *Netlist* stage of the design flow, replacing design primitives that introduce half-latches with primitives that use explicitly generated constants in the EDIF netlist. This approach is likely to succeed and should be reasonably easy to automate, but must be done carefully so that optimizations during technology mapping do not introduce half-latches.

The *Physical Database* and the *Bitstream* representations of the design are the most promising design representations for identifying the presence of half-latches and, possibly, for modifying designs to eliminate the dependence on half-latches since they can represent the final placed-and-routed forms of the designs. So, if the half-latches are removed at these points in the design flow, half-latch problems should not exist. With these latter design representations, the designer is dealing with the design expressed in terms of FPGA resources at a very low level of abstraction.

Modifying designs while in these low-level forms does have some disadvantages. First, only a handful of tools are useful for manipulating designs at this level. Second, because of the low abstraction level, a detailed knowledge of the devices is required to make the design modifications or design tools to automate these modifications. Third, because the designs have been placed and routed before being modified, there is a slight possibility that routing or logic resources may be too limited to remove the half-latches or that the modifications may impact timing if existing routing or logic resources must be disturbed to complete the operation. Finally, without some additional information, it may be difficult to remove a design's dependence on half-latches such that the logic redundancy techniques used to mitigate bitstream and user memory SEUs are still effective after half-latch mitigation. For instance, a tool that simply removes the dependence on half-latches for constants may unintentionally introduce additional SEU sensitivities by creating single points of failure across normally redundant circuit structures. To ensure the integrity of logic redundancy used for SEU mitigation, the half-latch mitigation approach must either be performed before logic redundancy is introduced or the approach must be aware of how the redundancy is used in the circuit and appropriately introduce redundancy into the constant sources.

As referenced in Figure 4, the options for detecting and mitigating half-latches at the *Physical Database* stage of the design flow for Xilinx FPGAs include *FPGA Editor* from Xilinx as well as third-party tools which modify the design while in the open Xilinx Design Language (XDL) format. *FPGA Editor* can be used to modify the design in the form of a Native Generic Database File (.ncd), a binary file generated by the Xilinx place-and-route tool that describes the final design before bitstream creation. Further, Xilinx provides a program called *xdl* that converts the proprietary Native Generic Database File format into the published, open XDL format. Once converted to XDL, third-party tools can be written to manipulate the XDL to create a modified design, which can then be converted back to the Native Generic Database format for bitstream creation.

At the *Bitstream* design stage, *JBits*—another Xilinx design tool—can be used for detecting and mitigating half-latches in designs. Currently not all of the resources in Virtex and Virtex-E devices are configurable via *JBits* (as of version 2.8) and the effects of half-latch mitigation on timing are not easily measurable when the design is modified as a bitstream. So, while theoretically feasible, a total solution at the *Bitstream* stage is not yet possible.

Of course, if the use of constants and, thus, half-latches were controllable in the synthesis and technology mapping stages via a switch or parameter of the tools, most, if not, all of the half-latch problems could be eliminated. Unfortunately, this problem is of relatively little interest to the commercial companies which provide the designs tools and is unlikely to be fixed by these tool companies.

B. RadDRC

To address the half-latch mitigation problem, we have created a tool called *RadDRC*, which parses the XDL representation of a design to locate half-latch issues and then can generate a new XDL design, automatically replacing half-latch constants with observable, repairable constants sources.

Versions 0.2.0 and 0.3.0 of the tool use the general approaches illustrated in (a) and (b) of Figure 3.

For the approach using externally generated constants, the latest version allows the externally generated constant to be either a “1” or a “0”. This constant is then routed to *all* internal resource inputs that require a constant value. The multiplexer (or mux) after the half-latch is configured as appropriate for the specific logic value that is needed at the resource input. For example, if a “0” is driven externally but a “1” value is needed by the resource, the mux is configured at the XDL design level to invert the incoming “0” to be a “1” at the input of the resource. When using a “0” from an external source, the half-latches driving the clock enables on the global clock buffers are still mitigated using LUT-generated “1” constants due to problems with the clock enable hardware in Virtex FPGAs.

As for the approach using LUTs to generate constants, instead of using a single source to generate a logic constant as with external constants, *RadDRC* uses a feature of the Xilinx *PAR* (place-and-route) tool to automatically allocate many LUTs and connect them to the user’s circuit as constant sources. The use of distributed, internal constant sources is superior to the use of external constant sources in that the distributed approach has no circuit-wide single points of failure for constant sources. Also, the approach does not require scarce I/O resources to perform the mitigation—only unused internal resources are utilized for constant sources.

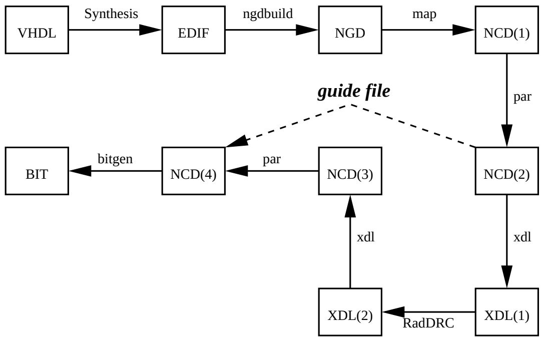


Fig. 5. Xilinx Design Flow when Using RadDRC 0.2 and 0.3

Figure 5 illustrates the overall FPGA tool flow which incorporates *RadDRC* to remove half-latches from designs. A design is run normally through the traditional Xilinx design flow from synthesis through the placement and routing tool, *PAR*, creating a completely placed-and-routed design as an NCD file (labeled *NCD(2)* in the figure). The placed-and-routed design is then converted to the XDL format, creating *XDL(1)* in Figure 5. The *RadDRC* tool is then executed on the XDL file to create a new XDL file, *XDL(2)*, which is a version of the design that does not use any critical half-latches but has a few unrouted nets. This half-latch mitigated design is then converted back to the NCD format (*NCD(3)*) to finish routing the design. The Xilinx *PAR* tool is re-executed on the design to route the constant sources to the inputs of the appropriate internal resources, producing *NCD(4)* in the process. To preserve timing and placement as much as possible from the original design, the original physical constraints file (PCF) can be used to constrain *PAR* to meet timing requirements and the original placed-and-routed NCD (*NCD(2)*) that met the timing constraints can be used as a guide file in the placement and routing process to constrain and reuse as much of the original routing and placement of the design as possible. Finally, a bitstream can be generated from *NCD(4)* to be used in the system.

Replacing half-latches in the manner described here has a few impacts on the design. First, it will use additional routing resources and, if a LUT-based constant-replacement strategy is used, additional logic resources. Second, theoretically, there should be no impact on design power or timing since the nets being introduced into the design do not change value. In practice, using the guide-file *PAR* approach to finish routing the designs, we have seen no noticeable impact on design speed due to half-latch SEU mitigation based on static timing analysis. Lastly, as mentioned before, half-latch replacement may impact logic redundancy techniques. As of this writing, *RadDRC* does not account for the use of TMR or other redundancy techniques and, thus, may introduce some SEU sensitivities into circuits which employ redundancy. This deficiency is solvable through a number of means. Until this is resolved, the tool still has great value in identifying existing half-latches and eliminating thousands of SEU sensitivities that are generally not detectable or correctable without taking a system off-line.

C. Half-latches and SEU Simulation

To help validate the results of the *RadDRC* tool before testing at the accelerator, the Virtex SEU simulator [9], created by Brigham Young University and Los Alamos National Laboratory, was used to test designs mitigated by the tool. Though the simulator injects faults into the bitstream and does not directly upset half-latches, a mechanism does exist which allows the SEU simulator to simulate upsets in half-latches as well—a result we were not expecting. As the simulator injects faults into the bitstream, it modifies the function of the design either in terms of logic functionality or routing. As the routing gets upset, the actions of connecting and disconnecting routing resources can apparently upset half-latches.

For instance, the half-latches which drive the clock-enable input muxes for slice flip-flops only require a two-bit bitstream corruption sequence to upset. The first bit connects a slice output mux to a hex routing line. The second bit connects the routing line to the clock-enable input mux through the half-latch. In this example, each upset is performed and then repaired so that only one bitstream error is present at any time. Once all of the bitstream errors have been repaired, the half-latch that normally holds the flip-flop clock enables high within the slice is upset, disabling the flip-flops until the half-latch is reset (through a full reconfiguration) or it recovers to its intended state.

The half-latch phenomenon has an interesting impact on SEU simulation, even for half-latch mitigated designs. As a simulation progresses, half-latches throughout the device are constantly being upset and reset through the course of the bitstream SEU simulation. This “background” or “hidden” state of the half-latches can affect whether or not a specific bitstream SEU causes an output error at a particular point in design execution. For instance, an upset might cause the input to a slice input mux to become unconnected, meaning that the half-latch is now driving the input to the slice input mux. If the disconnected signal's value matches the current value of the half-latch, it is possible that the SEU, which would regularly interrupt the normal operation of the circuit can be masked by the fact that the half-latch just happens to have the same value as the original signal. If the half-latch were to have the opposite value, the bitstream SEU could affect the operation of the design.

As it is currently implemented, the SEU simulator is not the perfect tool for testing a design's half-latch sensitivities. From our experiments, it is clear that, for a specific half-latch to be upset in a design, only a few very specific bitstream upset sequences can perform the half-latch upset through this indirect mechanism. To make the simulator a better test for half-latch sensitivities, the simulator would need to be aware of which specific bit-upset sequences are required to upset each half-latch. For now, the sequential and random upset modes appear to simulate half-latch upsets for slice resources well. On the other hand, while IOB half-latches still upset with these types of bitstream upset sequences, they are not as easily upset as slice half-latches.

IV. Radiation Experiment

Los Alamos National Laboratory and Brigham Young University conducted a proton radiation experiment at Crocker Nuclear Laboratory in November 2002 to validate our SEU simulator [10] and the *RadDRC* 0.2.0 tool. Regarding half-latches, the experiment's goal was to measure the average fluence until failure (FuF) under proton radiation for both a normal FPGA design and a version of the same design which had undergone *RadDRC*'s half-latch mitigation scheme. "Failure" in this case is when a design continues to operate improperly even when its configuration data is error-free and the design has been reset—the signature observed when a half-latch critical to design operation has been upset. This FuF quantity provides us with a measure for comparing the relative circuit reliability for the unmitigated and mitigated designs.

The experiments were conducted using 63.3 MeV protons with beam fluxes of $1.0x10^7$ and $3.5x10^7 \frac{p}{cm^2 \cdot s}$. The designs executed on a modified USC/ISI SLAAC-1V FPGA board with a free running system clock of 20 MHz for all but one trial, where a 2 MHz system clock was used. For each test, one copy of the design under test (DUT) executed in an FPGA which was exposed to proton radiation. A second, "golden" copy of the design executed synchronously with the DUT in an FPGA shielded from the protons. A third FPGA performed real-time output differencing between the two designs so we could quickly determine when the outputs for the two designs no longer matched. During the experiment, our software continuously identified and repaired configuration bitstream upsets and reset both designs when the two design's outputs differed, allowing their operation to be resynchronized—this sampling and repair process occurred at about a 2 Hz rate. When the DUT's output continued to differ from the output of the "golden" design despite having an error-free configuration bitstream and having been reset multiple times, a half-latch upset failure was identified. For this test, a single Xilinx XCV1000 FPGA (an XCV1000 FG680 AFP0017 F1102747A 5C) was used for the DUT FPGA. A more detailed explanation of the overall test setup can be found in [10].

With a limited amount of beam time for the test, we dedicated most of our time to study the SEU effects of proton radiation on a single design, the series of 8 72-bit multipliers followed by an adder tree illustrated in Figure 6. This data-path-intensive circuit was selected to represent a few of the operators and the architecture of some digital signal processing hardware. Due to limitations in our modified SLAAC1-V, the two inputs to the design are based on a single 36-bit, LFSR-generated input—one input being the original 36-bit input and the other being a permuted version of the same input. Further, due a limitation of 72 output bits, the final result had to be reduced by 3 bits to fit in the 72-bit output word. The design did not register I/O at the pads, so, in retrospect, it was not a good design to test I/O related half-latch effects. For the accelerator test, *RadDRC* version 0.2.0 was used to create a half-latch mitigated version of this design using the IOB-based mitigation approach illustrated in Figure 3(a).

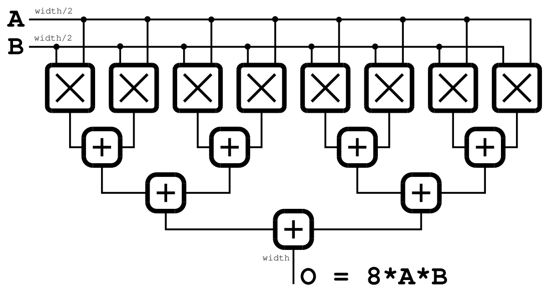


Fig. 6. Test design used in the half-latch mitigation test

Initially, we expected that observing 3 or 4 consecutive DUT output errors after bitstream repairs and design resets would be enough to identify the occurrence of a persistent design failure due to half-latch SEUs, assuming that half-latch upsets should not recover over time. As it turned out, this threshold for identifying half-latch failures was too low.

To begin with, the probability of 3 to 4 consecutive output errors due to SEUs in the bitstream and flip-flops without half-latch effects was small (about 1.5% out of the 4581 output error sequences observed for the mitigated design), but this probability would have been significant enough for even the half-latch mitigated design to have caused false detections. In general, bitstream and flip-flop SEUs never caused more than 5 consecutive output errors and they caused 5 consecutive errors only once out of the 4581 output error sequences observed for the half-latch mitigated design.

Further, we discovered that half-latches could recover after as few as 8 or 10 consecutive sample periods or after as many as 45 consecutive sample periods whether or not the proton beam was on. Figure 7 illustrates the recovery phenomenon for a design which was not mitigated for half-latches. The plot displays the number of consecutive output errors for a test run

TABLE II
FLUENCE UNTIL FAILURE FOR THE UNMITIGATED AND MITIGATED VERSIONS OF THE SAMPLE DESIGN

Design Test	Total Failures	Total Fluence (p/cm^2)	Ave Fluence to Failure (p/cm^2)	Accum. Dosage Range (krads)
Unmitigated Design				
Set 1	5	5.80E10	1.16E10	17.6 to 25.4
Set 2	15	5.42E10	3.62E9	57.2 to 64.5
Set 3	13	1.74E10	1.34E9	82.9 to 85.1
All	33	1.30E11	3.93E9	17.6 to 85.1
Mitigated Design				
All	1	4.10E11	4.1E10 to 1.93E12 w/ 90% conf.	10.7 to 86.5

over time. As mentioned before, the recovery of half-latches to their intended state is likely due to leakage in the *T3* transistor illustrated in Figure 1.

To ensure that half-latch SEUs were easily distinguishable and that as much half-latch data was collected as possible, designs were executed until the operators observed a significant degree of error persistence without design recovery even though the design had a good bitstream and had been reset. In practice, we discovered that thresholds greater than 50 were adequate for identifying persistent, half-latch-related failures.

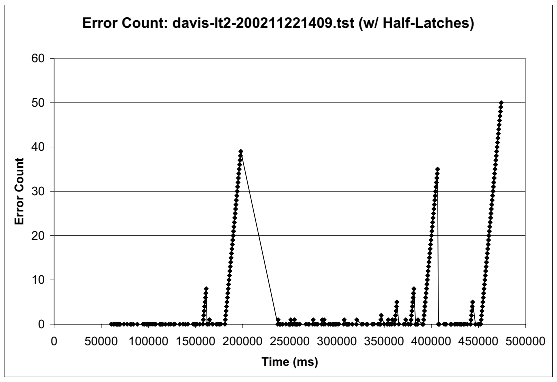


Fig. 7. Half-latch recovery while under proton radiation

Table II provides the observed average fluence-until-failure measurements for the sample design and its mitigated version. Note that a single failure was observed in the mitigated design due to a few half-latches that were missed by the *RadDRC* 0.2.0 tool. Though the observed "average" fluence to failure for the mitigated design was $4.10 \times 10^{11}\, p/cm^2$, the counting statistics error due to observing a single event dictates that the actual average is on the range of 4.1×10^{10} to $1.93 \times 10^{12}\, p/cm^2$ with a 90% confidence. Therefore, the actual improvement of the mitigated design over the unmitigated design is on the range of 10x to 470x despite an observed 100x improvement.

The above results only take into account the half-latch SEUs that caused output errors that did not recover quicky (within a few minutes). To take into account the more "transient" output errors caused by half-latches, the sampled output error data was analyzed for both designs to develop a tighter threshold for identifying these events. In the mitigated design, the consecutive error length distribution curve (having 4581 output error sequences) for this specific design was approximately Poisson with $\lambda = .13$. This would suggest that a threshold as tight as 6 consecutive errors (with a probability $<< .01\%$) should provide good results in distinguishing output errors due to bitstream or flip-flop SEUs from those due to half-latch SEUs. With a threshold of 6, a total of 52 critical half-latch SEUs were identified—57.6% more than with the higher threshold. Though this shows that the unmitigated design was affected by more half-latch SEUs than Table II reflects, the numbers do not greatly affect the fluence-until-failure comparisons.

During the analysis of fluence until failure, we discovered another interesting trend. For the unmitigated design, three distinct sets of data were taken at different points during the test and, hence, at three different ranges of accumulated dosage. As the FPGA accumulated dosage, the average fluence until failure due to half-latch SEUs for the unmitigated design decreased. Figure 8 shows this fluence-until-failure trend due to accumulated dosage with a linear fit for the trend. Note that the maximum recommended total ionizing dose for this part is 100 krads(Si) [5].

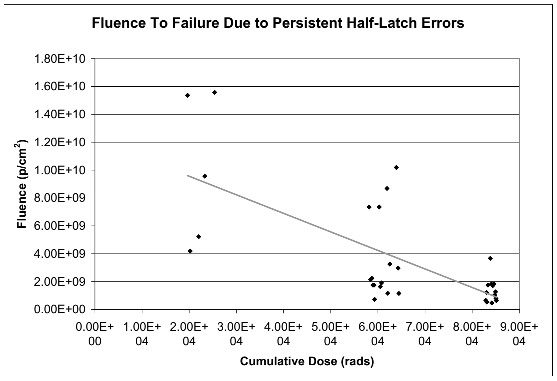


Fig. 8. The trend in fluence-until-failure due to persistent half-latch errors vs. accumulated dosage for the Virtex FPGA

In summary, the most significant finding is that the IOB-based half-latch mitigation used on the FPGA design had a significant impact on the reliability of the design, leading to at least an order of magnitude improvement in reliability, possibly two orders. Additionally, these levels of reliability improvement should increase as the half-latch mitigation tools approach perfect or near perfect mitigation of half-latches. Ideally, if a design is properly mitigated, there should be no persistent, half-latch-related errors in the design.

V. CONCLUSIONS

This paper has described, in some detail, the effects of radiation-induced upsets in the half-latch structures of Xilinx Virtex FPGAs and has illustrated that half-latches must be mitigated to ensure the reliable operation of FPGA designs in the space environment. Further, this paper has provided a survey of possible half-latch mitigation approaches and has presented the effectiveness of a specific half-latch mitigation solution which modifies a placed-and-routed design using of an externally generated constant.

Through the proton radiation experiment performed at Crocker Nuclear Laboratory, we saw more than an order of magnitude improvement in terms of average fluence until failure for a half-latch mitigated design over an unmitigated design when considering persistent half-latch upsets that caused output errors in the design. Ideally, if properly mitigated, there should be no persistent half-latch upsets that will affect a design, so the results of this experiment would have been better if the mitigation technique was perfectly applied. Despite the half-latches missed by the specific software we tested, we have demonstrated that the approach taken can significantly improve a design's reliability in the presence of radiation.

In addition to the improvement in fluence until failure we observed, we also discovered that half-latches may recover to their intended states over time after an SEU occurs. This effect is most likely due to leakage currents within the half-latch circuit itself. Despite this characteristic, the designer has no guarantee of when or if a half-latch will recover. Consequently, since half-latch SEUs are not easy to detect or correct, the designer is better off eliminating half-latches from the design rather than somehow depending on a scheme that expects half-latches to recover after some period of time.

Our future efforts will include the application of these same techniques to newer generations of Xilinx radiation-tolerant FPGAs based on the Virtex-II architecture. We also intend to improve the current *RadDRC* tool to properly mitigate half-latches in designs with logic redundancy so that the half-latch mitigation process itself does not introduce SEU sensitivities that reduce the efficacy of logic and routing redundancy in mitigating bitstream and user memory SEUs.

ACKNOWLEDGMENT

The authors would like to thank Xilinx and, specifically, Carl Carmichael and Joe Fabula for their support of this work. We would also like to thank the University of Southern California's Information Sciences Institute (USC/ISI) for their support of our SEU simulation work. Finally, we like to acknowledge the Department of Energy's funding of this work through the Deployable Adaptive Processing Systems project.

REFERENCES

[1] E. Fuller, M. Caffrey, P. Blain, C. Carmichael, N. Khalsa, and A. Salazar, "Radiation test results of the Virtex FPGA and ZBT SRAM for space based reconfigurable computing," in *MAPLD Proceedings*, September 1999.

[2] C. Carmichael, M. Caffrey, and A. Salazar, "Correcting single-event upsets through Virtex partial configuration," Xilinx Corporation, Tech. Rep., June 1, 2000, xAPP216 (v1.0).

[3] E. Fuller, M. Caffrey, A. Salazar, C. Carmichael, and J. Fabula, "Radiation testing update, SEU mitigation, and availability analysis of the Virtex FPGA for space reconfigurable computing," in *3rd Annual Conference on Military and Aerospace Programmable Logic Devices (MAPLD)*, 2000, p. P30.

[4] F. Lima, C. Carmichael, J. Fabula, R. Padovani, and R. Reis, "A fault injection analysis of Virtex FPGA TMR design methodology," in *Proceedings of the 6th European Conference on Radiation and its Effects on Components and Systems (RADECS 2001)*, 2001.

[5] C. Carmichael, E. Fuller, J. Fabula, and F. D. Lima, "Proton testing of SEU mitigation methods for the Virtex FPGA," in *4th Annual Conference on Military and Aerospace Programmable Logic Devices (MAPLD)*, 2001, p. P6.

[6] C. Carmichael, "Triple module redundancy design techniques for Virtex FPGAs," Xilinx Corporation, Tech. Rep., November 1, 2001, xAPP197 (v1.0).

[7] M. Caffrey, "A space-based reconfigurable radio," in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, T. P. Plaks and P. M. Athanas, Eds. CSREA Press, June 2002, pp. 49–53.

[8] P. Graham, M. Caffrey, M. Wirthlin, D. E. Johnson, and N. Rollins, "Reconfigurable computing in space: From current technology to reconfigurable systems-on-a-chip," in *Proceedings of the 2003 IEEE Aerospace Conference*. Big Sky, MT: IEEE, March 2003, pp. T07_0603.1–12.

[9] M. Wirthlin, E. Johnson, N. Rollins, M. Caffrey, and P. Graham, "The reliability of FPGA circuit designs in the presence of radiation induced configuration upsets," in *Proceedings of the 2003 IEEE Symposium on Field-Programmable Custom Computing Machines*, K. Pocek and J. Arnold, Eds., IEEE Computer Society. Napa, CA: IEEE Computer Society Press, April 2003, p. TBA.

[10] E. Johnson, M. Caffrey, P. Graham, N. Rollins, and M. Wirthlin, "Accelerator validation of an FPGA SEU simulator," *IEEE Transactions on Nuclear Science*, December 2003, submitted.

Paul Graham is a member of the technical staff at Los Alamos National Laboratory. His research interests include design automation tools for FPGA-based reconfigurable computing, applications of reconfigurable computing, and FPGA architecture. Dr. Graham obtained an M.S. degree in electrical engineering from Brigham Young University in 1996, and a Ph.D. in Electrical Engineering from Brigham Young University in 2001.

Michael Caffrey is a member of the technical staff at Los Alamos National Laboratory. His current work involves the production of a FPGA-based reconfigurable computing platform for space and has interests in reconfigurable computing and digital signal processing. Mr. Caffrey obtained an M.S. degree in electrical engineering from the University of New Mexico in 1995.

Michael Wirthlin is an assistant Professor at Brigham Young University in Provo, UT. His research interests include tools and applications for FPGA-based reconfigurable computing, power efficiency in FPGA designs, and synthesis techniques for digital systems. Dr. Wirthlin obtained a Ph.D. in electrical engineering from Brigham Young University in 1997.

D. Eric Johnson is a graduate student in the electrical and computer engineering program at Brigham Young University with interests in digital signal processing and reconfigurable computing.

Nathan Rollins is an undergraduate student in the electrical and computer engineering program at Brigham Young University with interests in reconfigurable computing.